# Project 1 - Working with text

## Project motivation

Imagine that we have a simple text file containing tabular information (for example, rows with individual cells separated by a space). Imagine we want to calculate the sum of all the values in the third column, or find their arithmetic mean, remove some column, or add a new one that will be the sum of the two previous cells? We need to run a cumbersome spreadsheet program (Excel, Libreoffice Calc, Google Sheets, etc.), import data into it, click what we want to do, and then save. And what if we have a lot of such files of the same type, but with different data? A simple command line tool would be useful to tell us what to do with a text file, and the tool would automatically run repeatedly over each file. It won't be pretty "clickable", but it will save us a lot of work.

## Project description

The aim of the project is to create a program that will implement the basic operations of spreadsheets. The input of the program will be text data, the input of operations will be through command line arguments and the program will output its result.

## Detailed specifications

Implement the program in the source file *sheet.c.* Input data will be read from standard input (stdin), output will be printed to standard output (stdout).

### Translation and submission of the source file

Upload: Upload the source file *sheet.c* through the information system.

Translation: Compile the program with the following arguments

```
$ gcc -std = c99 -Wall -Wextra -Werror sheet.c -o sheet
```

### Execution syntax

The program starts in the following form: (./sheet indicates the location and name of the program):

Adjust table size:

```
. / sheet [-d DELIM] [Table editing commands]
```

or data processing:

```
. / sheet [-d DELIM] [Row selection] [Data processing command]
```

- Argument -d specifies which characters can be interpreted as delimiters for individual cells. Each character in the input line that is contained in a DELIM string acts as a delimiter for two adjacent cells. By default, DELIM is a string containing a space. Multiple occurrences of the same characters in a DELIM string are ignored. The first character from the DELIM string will also be used as a delimiter for output values.

## Edit a table

Editing the table causes the table to be enlarged or reduced, resp. rows and columns. Multiple commands for editing a table can be entered. In this case, the following will be specified as a sequence of multiple command line arguments:

Table editing commands:

- irow R - inserts a table row before row R> 0 (insert-row).
- arow - appends a new table row to the end of the table (append-row).
- drow R - deletes row number R> 0 (delete-row).
- drows NM - removes lines N to M (N <= M). In the case of N = M, the command behaves the same asdrow N.
- icol C - inserts an empty column before the column given by the number C.
- acol - adds an empty column after the last column.
- dcol C - deletes column number C.
- dcols NM - removes columns N to M (N <= M). In the case of N = M, the command behaves the same asdcol N.

Separate commands and command sequences that do not have row or column collision numbers will be checked (eg arow irow 10, or dcol 5 drow 3).

## Data processing

Data processing is the modification of the contents of individual table cells. Each program run can contain at most one data processing command. The commands for data processing are as follows:

- Commands that are required for successful completion of the project:
  - cset C STR - the string STR will be set to the cell in column C.
  - tolower C - the string in column C will be converted to lowercase.
  - toupper C - the string in column C will be converted to uppercase.
  - round C - in column C, rounds the number to an integer.
  - int C - removes the decimal part of the number in column C.
  - copy NM - overwrites the contents of the cells in column M with the values in column N.
  - swap NM - swaps cell values in columns N and M.
  - move NM - moves column N before column M.
- Commands that are optional for successful completion of the project:
  - csum CNM - a cell representing the sum of cell values on the same row in columns N to M inclusive will be stored in the cell in column C (N <= M, C must not belong to the interval <N; M>).
  - cavg CNM - similar to csum, but the resulting value represents the arithmetic mean of the values.
  - cmin CNM - similar to csum, but the resulting value represents the smallest value found.
  - cmax CNM - similar to cmin, but this is the maximum value found.
  - ccount CNM - similar to csum, but the resulting value represents the number of non-empty values of the given cells.

- cseq NMB - inserts incrementally increasing numbers (by one) into the cells in columns N to M inclusive, starting with the value B.
- The meaning of the following statements in combination with row selection statements is undefined and is left to the implementation. The combination will not be tested.
- rseq CNMB - in column C, inserts ascending numbers starting with B into the cells of each row from row N to row M inclusive. The number M can be replaced by a hyphen. In this case, it means the last line of the file.
- rsum CNM - inserts the sum of cell values  in column C on rows N to M inclusive into the cell in column C on row M + 1.
- ravg CNM - similar to rsum, but the resulting value represents the arithmetic mean.
- rmin CNM - similar to rsum, but the resulting value represents the smallest value.
- rmax CNM - similar to rsum, but the resulting value represents the largest value.
- rcount CNM - similar to rsum, but the resulting value represents the number of non-empty values  of the given cells.

## Row selection

Data processing commands can be applied not only to the entire table, but only to selected rows. Commands for selecting such lines will be entered on the command line before the data processing commands:

- rows NM - the processor will process only rows N to M inclusive (N <= M). N = 1 means processing from the first line. If the character - (dash) is entered instead of the number M, it represents the last line of the input file. If the hyphen is also replaced by an N column, this means selecting only the last row. If this command is not specified, all lines are considered by default.
- beginswith C STR - the processor will process only those rows whose cell contents in column C begin with the string STR.
- contains C STR - the processor will process only those rows whose cells in column C contain the string STR.

At most one row selection command per program run will be checked. The line selection combination does not define the assignment and leaves it to the implementation.

## Implementation details

- The program outputs the resulting table. Input
- table cannot be an empty file.
- No command number identifying a row or column can be less than 1.
- The program assumes that the number of cells in the first row of the table represents the number of columns of the entire table. If there is a table on the input that does not meet this condition, the behavior of the program is undefined (however, the program must not end with a run-time error, such as a memory access error).
- If the table does not contain the number of rows or columns required in the specified commands, this fact is ignored: (i) the table modification commands will not be executed on the missing rows or columns, (ii) the data selection criteria above the missing rows or columns will not be met.

- In the case of a combination of line selection with commands rseq to rcount the data of those rows that do not match the given selection will not be counted (eg for the arithmetic mean, only those rows that match the specified criterion will be counted).
- The maximum supported length of a string in a cell or argument is 100. The maximum length of an entire row is 10KiB. For longer strings, the program warns with an error message and terminates with an error code.

**Limitations in the project**

It is forbidden to use the following functions:

- calls from the malloc and free families - working with dynamic memory is not needed in this project,
- calls from the family fopen, fclose, fscanf, ... - working with files (temporary) is not desirable in this project, calls
- qsort, lsearch, bsearch and hsearch - the goal is to think about algorithmization and data structure.
- calling the exit function - the goal of the project is to learn how to create program constructs that can handle an unexpected state of a program.

## Unexpected behavior

Respond to program runtime errors as usual: Respond to unexpected input data, input data format, or function call errors by interrupting the program with a brief and concise error message on the appropriate output and the corresponding return code. Reports will be in ASCII encoding in Czech or English.

## Examples of inputs and outputs

Phonebook help file:

```
$ cat tab1.txt
Exercises: Point: Project
First programs: 0:
Cycle, types: 0:
Chains: 0:
Function 1: 0:
Structures: 2:
Indicators: 2:
Function: 0: Defense 1
Malloc, dbg: 2:
: 0:
: 0:
Iteration, recursion: 2:
: 0: Defense 2
Dynamic structures: 2:
$ ./sheet -d: icol 1 <tab1.txt> tab1a.txt
$ ./sheet -d: rows 1 1 cset 1 Tyden <tab1a.txt> tab2.txt $ cat tab2.txt

Week: Tutorial: Item: Project: First
Programs: 0:
: Cycle, types: 0:
: Chains: 0:
: Function 1: 0:
: Structures: 2:
: Indicators: 2:
: Functions: 0: Defense 1
: Malloc, dbg: 2:
```

```
:: 0:
:: 0:
: Iterations, Recursion: 2:
:: 0: Defense 2
: Dynamic Structures: 2:
$ ./sheet -d: rseq 1 2 - 1 <tab2.txt> tab3.txt $ cat tab3.txt

Week: Tutorial: Item: Project 1: First
Programs: 0:
2: Cycle, types: 0:
3: Strings: 0:
4: Function 1: 0:
5: Structures: 2:
6: Indicators: 2:
7: Function: 0: Defense 1
8: Malloc, dbg: 2:
9 :: 0:
10 :: 0:
11: Iteration, recursion: 2:
12: 0: Defense 2
13: Dynamic structures: 2:
$ ./sheet -d: arow <tab3.txt> tab3a.txt
$ ./sheet -d: rows - - cset 2 "total points" <tab3a.txt> tab4.txt $ cat tab4.txt

Week: Tutorial: Item: Project 1: First
Programs: 0:
2: Cycle, types: 0:
3: Strings: 0:
4: Function 1: 0:
5: Structures: 2:
6: Indicators: 2:
7: Function: 0: Defense 1
8: Malloc, dbg: 2:
9 :: 0:
10 :: 0:
11: Iteration, recursion: 2:
12: 0: Defense 2
13: Dynamic structures: 2:: total
points ::
$ ./sheet -d: rsum 3 2 14 <tab4.txt> tab5.txt $ cat tab5.txt

Week: Tutorial: Item: Project 1: First
Programs: 0:
2: Cycle, types: 0:
3: Strings: 0:
4: Function 1: 0:
5: Structures: 2:
6: Indicators: 2:
7: Function: 0: Defense 1
8: Malloc, dbg: 2:
9 :: 0:
10 :: 0:
11: Iteration, recursion: 2:
12: 0: Defense 2
13: Dynamic structures: 2:: total
points: 10:
```

## Evaluation

The following factors have the main influence on the final evaluation:

- translatability of the source file,
- source file format (division, alignment, comments, appropriately chosen identifiers), decomposition of the
- problem into subproblems (suitable functions, suitable length of functions and function parameters), correct
- choice of data types, or creation of new types,
- correct functionality of editing and processing the table
- and handling of error states.

### Functionality priorities

1. Table editing commands (each program run accepts a maximum of one command).
2. Mandatory data processing commands.
3. Sequence of table edit commands (each program run accepts multiple table edit commands in multiple arguments).
4. Commands for row selection.
5. Selected optional data processing commands. The evaluation of this implementation will be beyond the maximum evaluation of the project and will be recognized only in the case of a smooth implementation of the mandatory parts.
6. Another premium solution (see below).

### Premium solution

The premium solution is voluntary and you can get bonus points for it. The condition for the award of premium points is an excellent elaboration of the obligatory parts of the assignment project. The final evaluation is fully in the competence of the teacher who will evaluate the project. The amount of premium points also depends on the sophistication of the solution. Examples of a premium solution can be:

- Implementation of optional identification of columns and ranges in the style of advanced spreadsheets - using letters: column 1 is marked with the letter A, the range of columns 26-28 with the string Z-AB. Combination of rows and cols as a commandrange A1: S42 (this is just an example). Command implementationconcatenate NM STR,
- which joins the strings of all the cells in columns N to M with a concatenation string STR, storing the result in column N and deleting all other columns up to column M. The string STR may be empty.

- Command implementation split ND, which divides the string in the cell in column N according to the delimiter D (one character) on several adjacent columns starting with N. All other columns will be moved to the right. It depends on the implementation how the program handles a variable number of columns in different rows.